UNIVERSIDAD AUTÓNOMA DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación

TRABAJO FIN DE GRADO

DETECCIÓN DE RITMO CARDIACO MEDIANTE VÍDEO.

Erik Velasco Salido.
Tutor: José María Martínez Sánchez

Junio 2015

DETECCIÓN DE RITMO CARDIACO MEDIANTE VÍDEO.

Erik Velasco Salido
Tutor: José María Martínez Sánchez

Video Processing and Understanding Lab
Departamento de Ingeniería Informática
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Junio 2015

Resumen

El objetivo de este trabajo fin de grado es diseñar y desarrollar un algoritmo que nos permita detectar el ritmo cardíaco mediante el análisis de secuencias de vídeo, tanto en secuencias naturales en color como en secuencias de máscaras de segmentación y de profundidad. El uso de estas últimas secuencias permite preservar la privacidad de la persona monitorizada y trabajar en condiciones de baja o nula iluminación.

Se ha diseñado e implementado una nueva aproximación, basada en un algoritmo base fundamentado en el trabajo previo en el que nos apoyamos. Una vez realizado y validado el algoritmo base sobre secuencias naturales en colores, se ha diseñado y desarrollado la nueva aproximación: un algoritmo que trabaja sobre secuencias de color, al igual que el algoritmo base, y que además trabaja sobre secuencias de máscaras de segmentación y profundidad proporcionadas por la cámara Kinect.

Se ha realizado un análisis del rendimiento de los distintos algoritmos en función de la distancia de la persona a la cámara con el fin de evaluar la viabilidad de un posible sistema combinado que utilice el nuevo algoritmo desarrollado.

Para la validación del algoritmo se ha grabado un conjunto de secuencias de vídeo (*dataset*), compuesto por secuencias de color, máscaras de segmentación y profundidad, además de la grabación del ritmo cardíaco medido por un pulsímetro. Con este *dataset* se ha podido obtener un completo conjunto de resultados en las distintas situaciones bajo estudio.

Palabras clave

ritmo cardíaco, detección, movimiento de la cabeza, seguimiento de puntos característicos, regiones de interés, análisis de componentes principales (principal component analysis - PCA), análisis de secuencias de vídeo, densidad espectral de potencia, secuencias de profundidad, máscaras de segmentación

Abstract

The objective of this Final Degree Thesis is to design and develop an algorithm that allows us to detect the heart rate by analyzing natural color, segmentation masks and depth video sequences. Using these last kind of sequences allows the preservation of the privacy of the monitored person and working in conditions of low or no lighting.

A new approach has been designed and implemented, starting from a base algorithm based on previous work in which we rely. After implementing and validating the base algorithm on color natural sequences, the new approach has been designed and developed: an algorithm working on color sequences, as the base algorithm, but also on masks segmentation and depth sequences provided by the Kinect camera.

We have analyzed the performance of different modalities depending on the distance of the camera to the person in order to assess the feasibility of a possible combined system using the various modalities supported by the proposed algorithm.

To validate the algorithm, a dataset has been recorded, composed of sequences natural color, segmentation masks and depth sequences, in addition to recording of the heart rate measured by a heart rate monitor. Using this dataset a complete set of results in the different situations under study has been obteined.

Keywords

Heart rate, detection, head motion, feature point tracker, region of interest, principal component analysis, video sequence analysis, power spectral density, depth video sequence, segmentation mask.

Agradecimientos

En primer lugar quiero dar las gracias a mi tutor, Chema, por haberme dado la posibilidad de realizar un trabajo tan interesante para mí como este, además de por su apoyo e implicación en todo momento.

También quiero dar las gracias mis compañeros del VPU, Pencho, Rafa, Diego, Marcos, María, Marta, gracias por vuestra ayuda.

No puedo olvidarme de todos mis compañeros durante estos años, pero especialmente a Paula, Ángel, Manu, Eduardo, Álvaro, Dani, Ana, Adrián Cobos, Carlos, Camilo gracias por todos los buenos momentos juntos.

No puedo olvidarme de dar las gracias a mis compañeros en Malta, Carlos, Faye y Marion. Gracias por vuestro apoyo durante este mes de mayo.

Mención aparte se merece mi compañero desde la UPM, Adrián Tomé, gracias por todos estos años juntos y espero que sean muchos más.

Un agradecimiento especial para mi compañera durante estos años tan duros para mí, gracias por apoyarme siempre cuando me venía abajo y por animarme a continuar adelante, gracias Marina.

Por último y no por ello menos importante, agradecer todo el apoyo que me ha prestado durante estos años mi familia sin ellos estas palabras no habrían sido posibles, y en especial a mis padres, por sus enseñanzas, sus sacrificios y esfuerzos para que yo haya podido realizar mis estudios, y alcanzar las metas que a ellos les fueron privadas en su momento por diferentes circunstancias.

Erik Velasco Salido

Junio 2015

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

1.1. Motivación

El ritmo cardíaco es uno de los signos vitales más importantes en el control médico de las personas. Los métodos actuales de medición de éste son invasivos para el paciente, ya que es necesario que la persona a la que se le está midiendo el ritmo cardíaco lleve algún tipo de sensor que realice esta evaluación.

Como se verá en el estado del arte, en la actualidad existen métodos para la obtención del ritmo cardíaco mediante el análisis de secuencias de vídeo en color a través de la observación de las variaciones de color producidas en la piel, generalmente en la región facial. Este tipo de aproximaciones, que son las más extendidas en la actualidad, tienen como mayores inconvenientes el no preservar la privacidad del individuo y el no funcionar en condiciones de baja o nula iluminación. Otra aproximación, menos popular, es el análisis del movimiento que provoca en la cabeza del sujeto bajo estudio el flujo sanguíneo a través de la aorta.

En este trabajo se busca diseñar, desarrollar y evaluar un algoritmo, que nos permita captar la actividad cardíaca mediante el análisis del movimiento inducido en la cabeza. Además de trabajar con secuencias naturales de color se hará uso de secuencias de vídeo de profundidad y de máscaras de segmentación que son tipos de secuencias que permiten preservar la privacidad. Además el uso de secuencias de profundidad tiene el valor añadido de permitirnos monitorizar situaciones en condiciones con baja iluminación o a oscuras..

Este tipo de aproximación nos permitirá estimar el pulso cardíaco sin la necesidad de utilizar dispositivos intrusivos para el sujeto, además de, en el caso de análisis a partir de grabaciones de profundidad, asegurar la preservación de la privacidad de las personas y permitir la grabación sin necesidad de iluminación (y seguir funcionando

en caso de que exista).

1.2. Objetivos

Para el desarrollo de este trabajo se plantean los siguientes objetivos:

- Estudio del estado del arte.
- Desarrollo de un conjunto de datos de evaluación (*dataset*) que incluya vídeos, así como las anotaciones del ritmo cardíaco obtenido con un pulsímetro, grabados a distintas distancias y en varias modalidades. Estará constituido por:
  - ✧ Secuencias grabadas con *webcam*, obteniendo secuencias naturales a color.
  - ✧ Secuencias grabadas con cámara Kinect, obteniendo tres tipos de secuencia sincronizadas: naturales a color, máscara de segmentación y profundidad.
- Desarrollo de una metodología de evaluación de resultados.
- Implementación del algoritmo base de análisis de movimiento para tener referencias.
- Mejora del algoritmo base para secuencias de color.
- Adaptación del algoritmo base a secuencias de profundidad y de máscaras de segmentación.
- Evaluación de la viabilidad del desarrollo de un sistema combinado en función de la distancia del sujeto a la cámara.

1.3. Organización de la memoria

La memoria consta de los siguientes capítulos:

- Capítulo 1. Motivación, objetivos y estructura de la memoria del trabajo.
- Capítulo 2. Estado del arte en la obtención del ritmo cardíaco mediante análisis de vídeo.
- Capítulo 3. Explicación del algoritmo base y del nuevo algoritmo propuesto para los diferentes tipos de secuencias.
- Capítulo 4. Descripción del *dataset* generado. Descripción de las pruebas realizadas y discusión sobre los resultados obtenidos.

- Capítulo 5. Conclusiones y trabajo futuro.
- Bibliografía.

Capítulo 2

Estado del arte

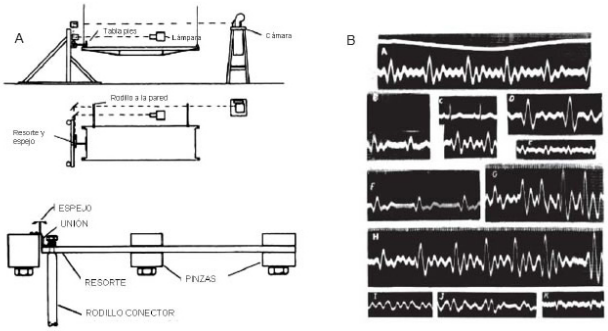
2.1. Introducción

Para la realización de este trabajo se ha estudiado diversa bibliografía relacionada con la temática del mismo. Durante este estudio se ha visto que existen diversas técnicas para la obtención del ritmo cardíaco mediante el análisis de vídeo, pudiéndose agrupar en dos grandes tipos: mediante variaciones de color y mediante movimiento.

Los métodos fundamentados en variaciones de color aprovechan los cambios de color producidos en la piel (generalmente de la región facial) para obtener el ritmo cardíaco, usando diversas técnicas que explicaremos más adelante en este capítulo. Por su parte, las técnicas basadas en movimiento analizan los movimientos imperceptibles inducidos por el flujo sanguíneo. Tanto los cambios de color como de movimiento inducidos por el flujo de la sangre son imperceptibles para el ojo humano, pero se han hecho visibles al ojo humano mediante técnicas de amplificación de vídeo [3] aplicadas a imperceptibles cambios de color y movimiento. Si bien es cierto que anteriormente ya había trabajos que se basaban en analizar cambios de color en vídeo, es a partir dicha publicación cuando se produce un incremento en la producción de trabajos basados en color y posteriormente fundamentados en el movimiento.

La idea de explotar la 3ª Ley de Newton para medir la actividad cardíaca no era un concepto nuevo, ya que sus orígenes se remontan a aproximadamente 1877, aunque fue en 1939 cuando Starr elaboró un balistocardiografo (ver figura 2.1), al que él también "bautizó", que inscribía las ondas en papel fotográfico. Para ello se colocaba al sujeto bajo estudio sobre una plataforma de baja fricción y el desplazamiento producido en la plataforma se usaba para medir el ritmo cardíaco. Hace unos años, He et al. [4] propusieron explotar el movimiento de la cabeza para monitorizar la actividad cardíaca mediante el uso de acelerómetros adheridos a la cabeza.

(a) Balistocardiógrafo de Nihon Kohden de 1953. Fuente: http://www.nihonkohden.com/company/history/1950s.html



(b) A. Balistocardiografo de Starr. B. Balistocardiogramas tomados por el Dr. Starr. Figura extraída de [5]

Figura 2.1: Ejemplos de balistocardiografos

2.2. Métodos de obtención del ritmo cardíaco por análisis de vídeo

La detección de ritmo cardíaco mediante análisis de vídeo es un campo en el que se lleva trabajando desde el año 2002 [6] con el uso de secuencias obtenidas mediante cámaras térmicas. En los últimos años ha experimentado un mayor crecimiento y han aparecido nuevas líneas de trabajo, como el uso de vídeos obtenidos mediante cámaras convencionales. Aunque ya se trabajaba con secuencias de color desde el año 2007 [7], es a partir de la publicación de un trabajo del Instituto Tecnológico de Massachusetts (MIT) [3], donde se mostraba que había una variación de color en el rostro producida por el flujo sanguíneo, cuando empieza a producirse este crecimiento. A raíz de este trabajo también se hacen visibles movimientos, que antes eran imperceptibles para el ojo humano, provocados por esta misma razón.

2.2.1. Mediante cambios de color

Este método es el más usado en la actualidad; durante el estudio del estado del arte se han consultado 18 referencias relativas a trabajos relacionados con esta aproximación. Esta técnica se basa en obtener las variaciones de color de los pixeles de la región facial, debidas a la circulación sanguínea, para posteriormente filtrarlas en la banda apropiada. A través de estas señales, analizadas directamente o procesadas, se obtiene una estimación del ritmo cardíaco. Para ello se busca la frecuencia a la que tenemos la máxima energía en un espacio transformado.

En la actualidad existen sistemas comerciales basados en esta tecnología, como por ejemplo, la aplicación producida por Philips [8] o la aplicación móvil Cardiio [9], disponible para descarga en las principales tiendas de apps.

(a) Philips Vital Signs Camera. Fuente: [8]

(b) Cardiio App. Fuente: [9]

Figura 2.2: Sistemas comerciales basados en cambios de color

Estos sistemas se basan en la selección de una región de interés, normalmente todo el rostro, determinada mediante algoritmos de detección facial, la gran mayoría basados en el algoritmo de Viola-Jones[10], aunque en algunos casos se usa la transformada de Haar entrenada para ello.

Dentro de esta metodología hay distintas variantes para medir los cambios de coloración:

- Realización de la operación AND de la imagen binarizada mediante la clasificación del tono de piel con el uso de la escala cromática de Fitzpatrick [11] y la imagen en el espacio YUV [12].
- Comparar frames en HSI [13].
- Separar en R, G y B, y normalizarlos, para posteriormente aplicar Análisis de Componentes Independientes (ICA - Independent Component Analysis) o Análisis de Componentes Principales (PCA - Principal Component Analysis) [14, 15, 16, 17, 1, 18, 19, 20].
- Separación en R, G y B y procesado posterior sin aplicar ICA o PCA [21, 22, 23].
- Obtención de la media de la intensidad o uso de banda G del rostro para después aplicar análisis espectral auto regresivo [24, 25, 26].
- Mediante el cálculo de las diferencias de brillo [7].
- Uso de descomposición piramidal y análisis de la componente R [27].

También tenemos distintas variantes a la hora de obtener la frecuencia dominante, siendo el cálculo de la Transformada Rápida de Fourier (FFT - Fast Fourier Transform) [15, 17, 1, 18, 19, 22] y el uso de la Densidad Espectral de Potencia (PSD - Power Spectral Density) [24, 16, 20, 25, 21] las dos técnicas más usadas.

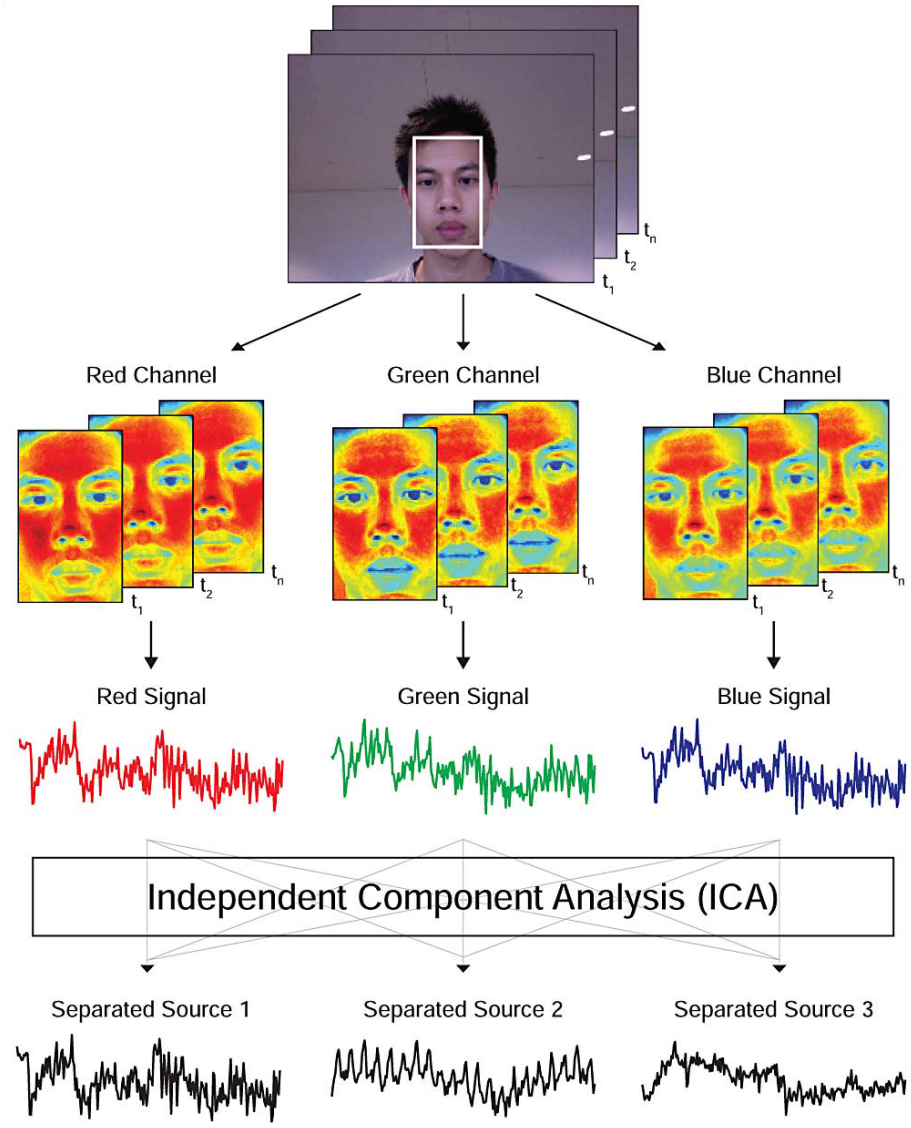


Figura 2.3: Ejemplo de algoritmo basado en cambios de color con separación en bandas de color y uso de ICA. Figura extraída de [1]

Estos sistemas necesitan que la piel de la zona de interés sea visible para la cámara. En contraposición las aproximaciones basadas en el movimiento no se ven limitadas por este aspecto.

Algoritmo	Detección facial	Cálculo cambio color	Obtención frecuencia dominante
Poh [17, 1, 18]	Viola Jones	Separación de componentes RGB	ICA + FFT
Pursche [19]	Viola Jones	Separación de componentes RGB	ICA + FFT
Scalise [20]	Manual	Separación de componentes RGB	ICA + PSD
Scully [25]	No usa región facial, utiliza huella dactilar	Solo banda G	PSD
Lewandowska [16]	Viola Jones	Separación de componentes RGB	PCA/ICA + PSD/Zero Crossing
Hsu [14]	Omron Okao	Separación de componentes RGB	ICA + SVR
Kwon [15]	Viola Jones	Separación de componentes RGB	ICA + FFT
Kleiner [13]	Detección de piel	HSI	Aplicación de distribución normal sobre 1ª y 2ª derivada
Carvalho [27]	Viola Jones	Descomposición piramidal y análisis de banda R.	Detección de picos.
Bolkhovsky [24]	No usa región facial, utiliza huella dactilar	Solo banda G	PSD
Bousefsaf [12]	Viola Jones	Skin AND LUV	Wavelets
Takano [7]	Manual	Cambios de brillo	Análisis espectral auto regresivo
Tarassenko [26]	Everinghan et al.	Solo banda G	Análsis espectral auto regresivo
Tsouri [21]	Haar	Separación de componentes RGB	cICA + PSD
Verkruyse [22]	Manual	Separación de componentes RGB	FFT
Wei [23]	Haar	Separación de componentes RGB	Laplacian Eigenmaps

Tabla 2.1: Tabla resumen de algoritmos basados en cambios de color

2.2.2. Mediante movimiento

Esta metodología es una alternativa a la extracción de la frecuencia del pulso a través del análisis de los cambios de color en la piel producidos por la circulación de la sangre.

En 2013 se realizaba, por parte de Balakrishnan et al. [28] en el Instituto Tecnológico de Massachusetts (MIT), el primer trabajo que explotaba el movimiento producido por la circulación sanguínea en la cabeza para obtener la actividad cardíaca mediante el análisis de vídeo. En este trabajo se usaba la detección de la región facial mediante el algoritmo de Viola-Jones para establecer la región de interés. Posteriormente se obtienen los puntos característicos, haciendo uso del algoritmo de Good Features to Track [29] contenidos en esa región y se realiza su seguimiento mediante el algoritmo KLT (Kanade Lucas Tracking) [30]. Una vez se obtiene la trayectoria de estos puntos se le aplica un filtrado en la banda de frecuencias adecuada. Una vez filtrada la señal se procede al Análisis de Componentes Principales y sobre el resultado de éste se calcula la Transformada Discreta de Fourier para obtener el ritmo cardíaco.

Posteriormente en 2014, en la Universidad de Aalborg, Irani et al. [31] replicaban el algoritmo realizado en el MIT y proponían modificaciones para obtener mejoras en grabaciones con movimiento voluntario de la cabeza. Los cambios introducidos en este trabajo son: el uso de un filtro de suavizado después del filtrado de la señal, y el uso de la Transformada Discreta del Coseno en lugar de la DFT.

Estos dos trabajos son las únicas referencias que hemos encontrado de sistemas que exploten este método para la obtención de la actividad cardíaca.

2.3. Conclusiones

Por todo lo comentado anteriormente en este capítulo, se ha decido trabajar sobre la tecnología para hallar el pulso mediante el análisis de movimiento en secuencias de vídeo, ya que se ha visto que es una forma novedosa y menos estudiada hasta el momento.

Además el movimiento se puede evaluar sobre otro tipo de imágenes no a color, como por ejemplo profundidad o con imágenes procesadas (ej. siluetas). Por lo tanto, éste tipo de técnicas generan como valor añadido la preservación de la privacidad del sujeto bajo estudio frente al análisis en secuencias de color y la independencia de las condiciones de iluminación para una correcta medición en el caso de secuencias de profundidad.

Capítulo 3

Diseño y desarrollo

3.1. Introducción

Una vez se ha analizado el estado del arte, se ha decidido realizar una reproducción del algoritmo del MIT [28], que será el algoritmo base que se utilizará como referencia en el análisis comparativo de los resultados obtenidos. En este capítulo se ha expuesto como se ha llevado a cabo el diseño y desarrollo de los distintos algoritmos, y el porqué de cada uno de ellos.

Primeramente se ha implementado el algoritmo base; una vez realizado éste, se ha mejorado dando lugar al algoritmo propuesto, con el que se ha obtenido una mejora de los resultados en las secuencias de color. El algoritmo desarrollado puede trabajar con secuencias de profundidad y con máscaras de segmentación que permiten preservar la privacidad, además en el primer caso se tiene un nuevo valor añadido al independizarse la captación de imágenes de las condiciones de iluminación y añade la posibilidad de analizar el movimiento en la 3ª dimensión.

3.2. Algoritmo base

3.2.1. Diseño

Este algoritmo base es una reproducción lo más fiel posible del algoritmo desarrollado por Balkrishnan en el MIT [28], aunque puede haber partes que no estén implementadas de igual forma por no describir el artículo suficientes detalles de cómo se implementaban.

Como se ha visto en el estado del arte, este algoritmo se basa en la obtención del pulso mediante el movimiento inducido por el flujo sanguíneo en la cabeza. Para ello, se ha seguido el esquema de la Figura 3.1. Primeramente se obtiene la región

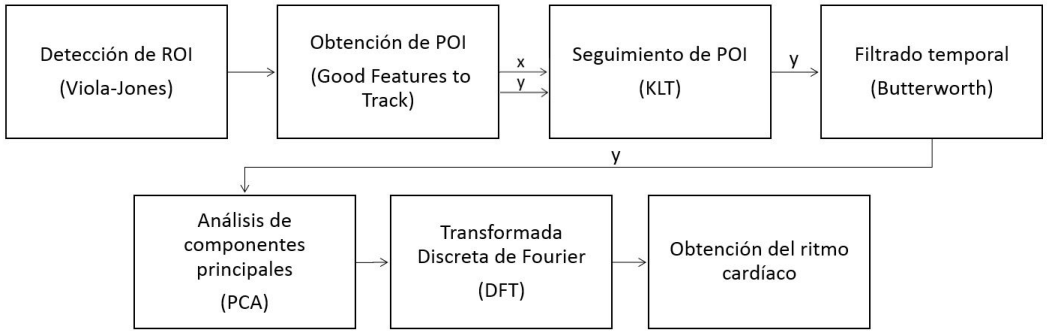


Figura 3.1: Diagrama algoritmo base

de interés mediante el uso del algoritmo de Viola-Jones [10], para posteriormente obtener los puntos característicos mediante el uso de Good Features to Track [29]. El algoritmo de Good Features to Track, basado en el algoritmo de detección de esquinas de Shi-Tomasi, calcula directamente el $min(\lambda_1, \lambda_2)$ y además trabaja bajo ciertas suposiciones que hacen que los puntos detectados sean más estables para su seguimiento.

Sobre los puntos hallados se realiza el seguimiento de su trayectoria con KLT [30]. Las señales que se obtiene como resultado del seguimiento se ha filtran, con un filtro paso banda Butterworth de orden 5 y banda de paso entre 0.75 Hz y 5 Hz, y se les aplica el análisis de componentes principales (PCA) quedando las componentes en el rango comprendido entre la segunda y la quinta. Para finalizar se realiza la Transformada Discreta de Fourier (DFT), sobre ésta se calcula la energía para buscar la frecuencia a la que se obtiene el máximo y con el resultado de ésta se calcula el ritmo cardíaco.

3.2.2. Desarrollo

El algoritmo se ha implementado íntegramente en Matlab. Para ello se han desarrollado un conjunto de funciones que componen el algoritmo como se puede ver en el pseudocódigo aquí expuesto. En la función principal del algoritmo base tenemos dos opciones de ejecución. La primera es sin hacer uso de enventanado; de esta forma procesaríamos todo el vídeo y obtendríamos el ritmo cardíaco medio de toda la secuencia. La segunda opción es usar enventanado; si elegimos esta opción debemos determinar el tamaño de la ventana a usar, en esta forma de ejecución obtenemos el pulso medio medido en cada ventana, las ventanas se van desplazando cada segundo.

```
%% Main algoritmo base
%% Entrada: video - video; use_window - parámetro que determina si
   usamos enventanado; t_window - tamaño de la ventana en segundos
%% Salida: HR - ritmo cardíaco medio medido por el algoritmo
function [HR]=algoritmo_base(video,use_window,t_window)
        orden=5; fl=0.75;     fh=5;     fs=30;   % Establecemos los
           parámetros del filtro
        NFFT=8192;
        vidRGB=lectura_fichero(nombreFichero,nFrames); % Función
           que lee la secuencia a procesar.
        [ROIAdap]=findROI(video); % Función que busca las ROI.
        [POI]=findPOI(ROIAdap,video); % Función encargada de la
           busqueda de los POI.
        [POI_Track]=KLT(POI, video); % Función que realiza el
           seguimiento de los POI.
        [pointT_filter] = filter_butterworth(POI_Track, orden, fl,
           fh, fs); % Función que filtra las trayectorias de los
           puntos obtenidas anteriormente.
        [PCA_results]=calc_pca(pointT_filter); % Función que
           realiza el PCA.
        [DFT_results]=calc_DFT(PCA_results, NFFT); % Función que
           realiza el cálculo de la DFT.
        [HR]=find_HR(DFT_results,NFFT,fs); % Función que nos
           devuelve el ritmo cardíaco.
end
```

La función *findROI* es la encargada de la detección de la ROI hace uso del algoritmo de Viola-Jones siguiendo lo expuesto por Balakrishnan [28]. En este caso se ha usado la función *vision.CascadeObjectDetector* que nos proporciona Matlab para la búsqueda de la cara. Una vez se obtiene la región facial en el primer *frame* de cada ventana, se ajusta para obtener las dos ROI como se explica en el trabajo de Balakrishnan [28]. En el algoritmo base éstas son, la región de la frente y la región que se encuentra debajo de la nariz, como se pueden ver en la Figura 3.2. Estas regiones se han elegido porque son las más estables en una secuencia colaborativa del sujeto, en la que el mismo está posando. Como se observa en la Figura 3.2, se ha eliminado la zona de los ojos; esto se realiza conforme a [28] con el objetivo de que los artefactos causados por el parpadeo no nos afecten. En los caso en los que la cara no es visible se han de seleccionar las regiones de interés de forma manual para el primer *frame* de cada ventana.

Figura 3.2: Regiones de interés

```
%% Función para la obtención de las ROI
%% Entrada: video - video
%% Salida: ROIAdap - matriz con las dos ROI
function [ROIAdap]=findROI(video);
        faceDetector=vision.CascadeObjectDetector(); % Implementación de Matlab para hacer uso de Viola-Jones.
        Region = step(faceDetector, objectFrame); % Obtenemos la ROI de la cara con Viola-Jones.
        ROIAdap=AdaptROIBase(Region); % Función que nos devuelve las dos secciónes de interés en una matriz.
end
```

Para la detección de los puntos característicos se ha implementado la función *findPOI* que calcule estos puntos haciendo uso de la función de Matlab *vision.CornerDetector*, basada en Good Features to Track[29] siguiendo lo expuesto en [28]. En esta función se buscan los puntos más significativos (esquinas), ubicados en el interior de las regiones de interés que se obtienen en el paso anterior. Esta función nos devuelve las coordenadas x e y de cada uno de los puntos de interés detectados.

```
%% Función para la búsqueda de los POI
%% Entrada: video - secuencia de video; RegionAdap - Matriz con las dos regiones de interés
%% Salida: POI - puntos de interés obtenidos
function [POI]=findPOI(ROIAdap,video);
        CornerDetector = vision.CornerDetector('Method','Minimum eigenvalue (Shi & Tomasi)');
```

Figura 3.3: Puntos característicos extraídos con algoritmo Good Features to Track

```
6           foreheadPoints = step(CornerDetector, rgb2gray(imcrop(
                objectFrame, RegionAdap(1,:)))); % Llamada a la función
                 para la detección de los puntos de interés
7           subnosePoints = step(CornerDetector, rgb2gray(imcrop(
                objectFrame, RegionAdap(2,:)))); % Llamada a la función
                 para la detección de los puntos de interés
8           POI=AdaptPoints(foreheadPoints, subnosePoints); % Llamada a
                 la función que nos adapta los puntos obtenidos en las
                regiones de interés y nos devuelve todos los puntos en
                una sola matriz.
9   end
```

En la función *KLT* se hace uso del algoritmo KLT [30] como en los casos anteriores, usando la implementación propia de Matlab *vision.PointTracker*, y nuevamente siguiendo el trabajo de Balakrishnan[28], se hace el seguimiento de todos estos puntos obteniendo su variación a lo largo del tiempo en ambas dimensiones (x(t), y(t)). Solamente se ha usado la variación en la componente en la dimensión y atendiendo a lo mencionado en [28] y se desecha la componente x.

```
1   %% Función para el tracking de los POI
2   %% Entrada: video - secuencia video; POI - puntos de interés a
        seguir
3   %% Salida: POI_Track - Trayectoria de los puntos seguidos.
4   function [POI_Track]=KLT(POI, video);
5           pointTracker= vision.PointTracker('MaxBidirectionalError',
                2); % Configuración del algoritmo KLT
6           initialize(pointTracker, Points, objectFrame); %
                Inicialización del tracker
```

```
7           for i=2:size(video,2)  % Bucle para seguimiento de los
               puntos
8                [Points] = step(pointTracker, frame);  % Llamada al
                   algoritmo KLT
9                POI_Track=component_Y(Points);  % Función que nos
                   deja solamente la componente y de los puntos
10          end
11  end
```

Una vez se han obtenido los vectores que contienen la variación temporal de los distintos puntos, éstos se han pasado a la función *filter_ butterworth*. En ella se ha usado un filtro paso-banda Butterworth de orden 5 con frecuencias de corte comprendidas entre 0.75 y 5 Hz correspondientes al intervalo de 45 a 300 pulsaciones de igual forma que se hace en [28]. Con este filtrado se eliminan posibles variaciones producidas por movimientos externos y trayectorias erráticas.

```
1   %% Función que filtra la trayectoria de los puntos de interés
2   %% Entrada: POI_Track - trayectoria de los puntos seguidos; fs -
       frecuencia de muestreo
3   %% Salida: pointT_filter - trayectoria filtrada
4   function [pointT_filter] = filter_butterworth(POI_Track, orden, fl,
        fh, fs)
5           d = fdesign.bandpass('N,F3dB1,F3dB2', orden, fl, fh, fs);
6           Hd = design(d,'butter');  % Generamos un filtro Butterworth
               paso banda de orden 5 con banda de pasa de 0.75 a 5 HZ
7           pointT_filter=filter(Hd,POI_Track);  % Filtramos las
               trayectorias con el filtro Butterworth
8
9   end
```

El resultado de las variaciones filtradas ha sido usado por la función *calc_ pca* encargada de realizar el Análisis de Componentes Principales. Para ello, se ha hecho uso de la función *PCA* de Matlab, que nos devuelve las componentes principales. Esto se ha hecho para descomponer el movimiento de los puntos característicos en subseñales para aislar el pulso. En este caso solamente se han elegido las componentes de la segunda a la quinta como se nos indica en [28].

```
1   %% Función que realiza el PCA
2   %% Entrada: pointT_filter - trayectoria filtrada de los puntos
3   %% Salida: PCA_results - ritmo cardíaco obtenido con el algoritmo
4   function [PCA_results]=calc_pca(pointT_filter);
```

```
5           ev=pca(pointT_filtered,'NumComponents',5); % Aplicamos PCA
                sobre la matriz de trayectorias para obtener los
                autovectores de las 5 primeras componentes.
6           signals=pointT_filtered*ev; % Calculamos las señales
                resultantes de aplicar el PCA a la matriz pointT_filter
                .
7           PCA_results=signals(2:5); % Nos quedamos con las
                componentes 2ª, 3ª, 4ª y 5ª.
8   end
```

Haciendo uso de la función *calc_ DFT* se ha calculado la DFT, con la función de Matlab, de las 4 componentes halladas anteriormente.

```
1   %% Función el cálculo de la DFT
2   %% Entrada: PCA_results - trayectoria filtrada de los puntos; NFFT
        - número de puntos para el calculo de la DFT
3   %% Salida: DFT_results - Matriz con las DFT de las cuatro
        componentes
4   function [DFT_results]=calc_DFT(PCA_results, NFFT);
5           for i=2:5
6                   DFT_results(i)=fft(PCA_results(i),NFFT)/(NFFT-1); %
                        Calculamos la DFT de las componentes 2 a 5
7           end
8   end
```

Con el uso de la función *find_ HR* obtenemos el ritmo cardíaco, mediante la búsqueda de la mayor energía. Para ello se ha calculado la energía de cada una de las DFT y se ha buscado el máximo absoluto de entre todas para así obtener la frecuencia dominante, que será la que se use para obtener el ritmo cardíaco. Para la obtención del pulso en pulsaciones por minuto, se multiplica la frecuencia adquirida por 60 segundos/minuto, obteniendo así el ritmo cardíaco, como en [28].

```
1   %% Función que nos devuelve el ritmo cardíaco
2   %% Entrada: DFT_results - DFT de las 4 componentes resultantes del
        PCA; NFFT - Número de puntos para el calculo de la DFT; fs -
        Frecuencia de muestreo
3   %% Salida: HR - ritmo cardíaco obtenido con el algoritmo
4   function [HR]=find_HR(DFT_results,NFFT,fs);
5           f_max=find_max_energy(DFT_results, fs, NFFT); % Llamamos a
                la función que nos devuelve la frecuencia a la cual
                tenemos el máximo de energía, sus argumentos de entrada
```

```
            son las DFT calculadas, la frecuencia de muestreo y el
            número de puntos de la DFT.
        HR=f_max*60; % Obtenemos el ritmo cardíaco multiplicando la
            frecuencia resultante por 60 segundos/minuto.
end
```

3.3. Algoritmo propuesto

3.3.1. Diseño

Nuestro objetivo final con el algoritmo propuesto es que pueda trabajar con secuencias de profundidad, para ello sobre el algoritmo base se han realizado varias modificaciones: selección de la ROI, uso de ambas coordenadas (x e y) de la trayectoria, y cálculo del ritmo cardíaco mediante la densidad espectral de potencia. Con estas modificaciones se ha obtenido un algoritmo que puede trabajar tanto con secuencias de color, como con secuencias de profundidad y máscaras de segmentación.

El uso de secuencias de profundidad viene motivado por la posibilidad de mantener la privacidad del sujeto, la robustez a condiciones de iluminación y el permitir medir las variaciones en la componente z.

Con la modificación de las selecciones de la ROI se ha buscado escoger las regiones más estables del rostro. Como se ha visto, en el algoritmo base se usan la región de la frente y la zona situada debajo de la nariz. En esta segunda zona, puede haber movimientos ocasionados porque el sujeto esté hablando o por un cambio gestual. Por ello se ha realizado la modificación de esta región y se ha escogido en su lugar la sección comprendida entre la frente y la parte inferior de la nariz. Para las secuencias de profundidad y máscaras de segmentación que nos proporciona Kinect, al no ser posible aplicar a estas imágenes un algoritmo de detección facial, se ha optado por realizar en esta implementación, por ahora, la selección de la ROI de forma manual por el usuario.

Otra variación es el uso de la coordenada horizontal que se obtiene del seguimiento de los puntos característicos. Como se ha comentado anteriormente el algoritmo original solamente hace uso de la componente vertical de la trayectoria y ahora se hace uso tanto de la componente vertical como la horizontal. El medir las variaciones en la componente z a priori se puede realizar con las secuencias de profundidad, pero una vez se analizó la precisión de las secuencias de profundidad que se obtenían mediante el uso de la cámara Kinect, se vio que no era realizable pues las variaciones

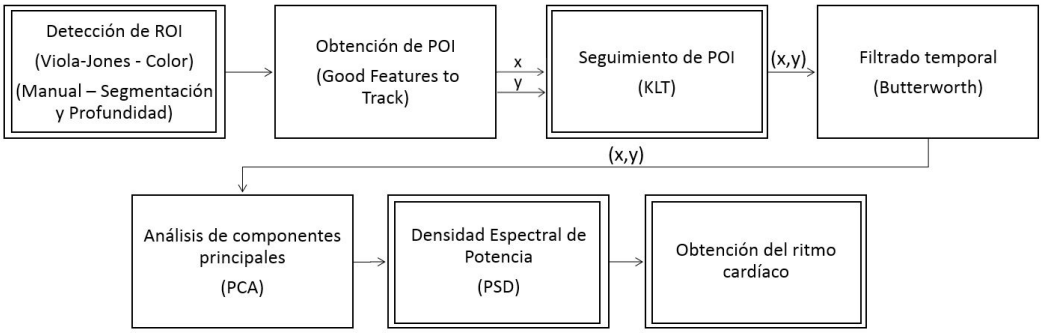


Figura 3.4: Diagrama algoritmo propuesto

son demasiado pequeñas para la precisión de la cámara[1]. También se planteó el uso de coordenadas polares, pero finalmente se descartó por sus resultados.

El último cambio es el uso de la densidad espectral de potencia, en lugar de la Transformada Discreta de Fourier. Se ha optado por probar la densidad espectral de potencia, ya que es una alternativa bastante usada en los trabajos basados en cambios de color estudiados en el estado del arte [24, 16, 20, 25, 21].

3.3.2. Desarrollo

En este apartado comentaremos todas las modificaciones que se han realizado sobre el algoritmo base. A continuación se puede observar el pseudocódigo de la función principal del algoritmo del propuesto. Como en el caso del algoritmo base podemos ejecutar nuestro algoritmo haciendo uso del enventanado o prescindiendo de él.

```
1 %% Main algoritmo color
2 %% Main algoritmo base
3 %% Entrada: video - video; use_window - parámetro que determina si
     usamos enventanado; t_window - tamaño de la ventana en segundos
4 %% Salida: HR - ritmo cardíaco medido por el algoritmo
5 function [HR]=algoritmo_propuesto(video,use_window,t_window)
6         orden=5; fl=0.75;     fh=5;     fs=30;   % Establecemos los
             parámetros del filtro
7         NFFT=8192;
8         vidRGB=lectura_fichero(nombreFichero,nFrames); % Función
             que lee la secuencia a procesar
```

[1]Habría que evaluar si con la nueva versión de la cámara Kinect tendríamos la precisión suficiente para explotar esta dimensión.

```
9           [ROIAdap]=findROI_new(video, type_seq); % Función que busca
                las ROI.
10          [POI]=findPOI(ROIAdap,video); % Función encargada de la
               busqueda de los POI.
11          [POI_Track]=KLT_xy(POI, video); % Función que realiza el
               seguimiento de los POI.
12          [pointT_filter] = filter_butterworth(POI_Track, orden, fl,
               fh, fs); % Función que filtra las trayectorias de los
               puntos obtenidas anteriormente.
13          [PCA_results]=calc_pca(pointT_filter); % Función que
               realiza el PCA.
14          [PSD_results]=calc_PSD(PCA_results, NFFT); % Función que
               realiza el cálculo de la PSD.
15          [HR]=find_HR_new(PSD_results,NFFT,fs); % Función que nos
               devuelve el ritmo cardíaco.
16  end
```

Primeramente se ha modificado la función *findROI* del algoritmo base dando lugar a la nueva función *findROI_ new* que se ha desarrollado para la obtención de las regiones de interés, este cambio se ha realizado por las razones mencionadas en el apartado de diseño. Para ello se hace uso también de la función *vision.CascadeObjectDetector* de Viola-Jones [10] de Matlab, pero se ha cambiado la adaptación de las ROI. En este caso se han probado diversas alternativas, como el uso de una única región de interés, o la modificación que finalmente se ha implementado en el algoritmo, por ser la opción más estable. En la Figura 3.5 podemos ver la comparativa entre las regiones de interés usadas en el algoritmo base y la alternativa propuesta.

Para el caso de las secuencias que contienen información de segmentación o de profundidad, se ha tenido que adaptar la selección de la región de interés. En esta clase de secuencias nos encontramos ante imágenes con la silueta segmentada sobre la que no se puede aplicar el algoritmo de Viola-Jones para detección de caras. Por lo que se ha optado por implementar dentro de la función *findROI_ new* la selección de forma manual la ROI para estas secuencias. En este tipo de secuencias se elige una sola región de interés, como puede verse en la Figura 3.6 , que abarca toda la cabeza del sujeto.

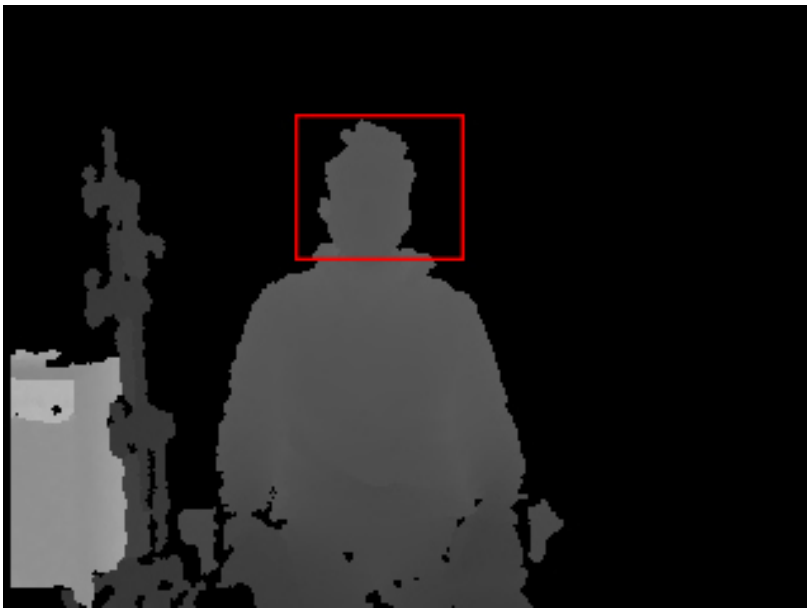
En la función para la búsqueda de puntos característicos (ver figura 3.7), se han probado otros algoritmos y tipos de puntos (como SIFT, SURF, etc.), pero no se han obtenido mejoras significativas en los resultados, por lo que se ha optado por mantener la misma función (*findPOI*) que en el algoritmo base de [28]. Para este tipo de secuencias podemos observar, Sub-Figuras 3.7b y 3.7c, como los puntos de interés

(a) Regiones de interés algoritmo base

(b) Regiones de interés algoritmo propuesto

Figura 3.5: Comparativa regiones de interés para secuencias de color

(a) Secuencia segmentada



(b) Secuencia de profundidad

Figura 3.6: Región de interés seleccionada manualmente

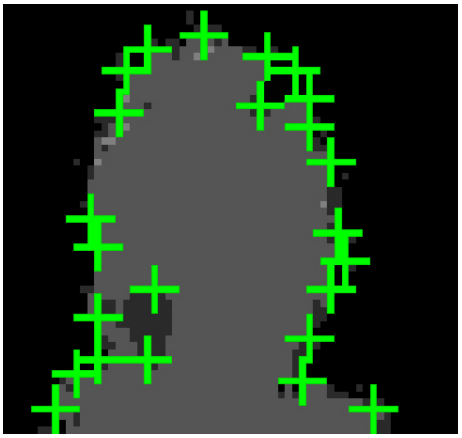
```
%% Función para la obtención de las ROI.
%% Entrada: video - secuencia de video; type_seq - tipo de
   secuencia
%% Salida: ROIAdap - matriz con las dos ROI para el caso de color o
    con la ROI en el caso de máscaras de segmentación o
   profundidad
function [ROIAdap] = findROI_new(video, type_seq)
        if strcmp(type_seq,'Colour')==1
                faceDetector=vision.CascadeObjectDetector();
                Region = step(faceDetector, objectFrame); %
                   Obtenemos la ROI de la cara con Viola-Jones.
                ROIAdap=AdaptROI_new(Region); % Función que nos
                   devuelve las dos secciónes de interés en una
                   matriz.
        elseif strcmp(type_seq, 'Depth')==1 || strcmp(type_seq, '
           Segmentation')==1
                imshow(vid(1).frame);
                [xF,yF] = ginput(2);
                ROIAdap = build_ROI(xF,yF); % Función que nos
                   devuelve la ROI.
        end
end
```

están situados en los bordes de la silueta y pueden aparecer algunos puntos en el interior por la baja resolución de la cámara.

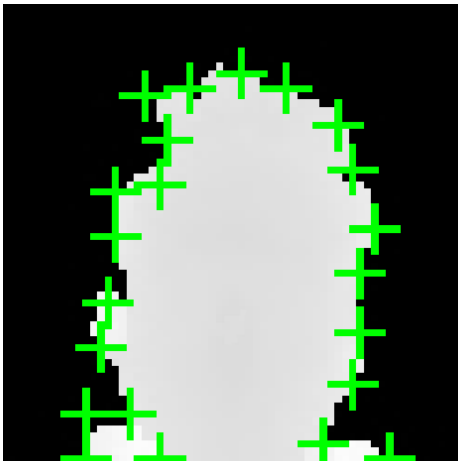
Como se ha mencionado en el apartado de diseño, se hace uso de ambas coordenadas, tanto x como y, frente al algoritmo base [28] que solamente hace uso de la trayectoria en la componente vertical. Este cambio viene motivado porque se ha observado que el movimiento inducido por el flujo sanguíneo tiene componentes en las 3 dimensiones. Además, en situaciones en las que la cabeza del sujeto no está totalmente vertical, el tener en cuenta la componente horizontal nos ayuda en la obtención del pulso.

Para realizar este cambio se ha implementado al función KLT_xy, que como podemos ver es la función KLT modificada para que nos devuelva las dos componentes x e y concatenadas, obtenidas mediante el uso del algoritmo KLT de Matlab *vision.PointTracker*, en un mismo vector para su posterior filtrado y análisis. El tener ambas componentes en un mismo vector nos ayuda a no tener que modificar el resto de funciones ya que este vector de componentes es igual al vector que contenía solamente las componentes en y, esto se puede hacer gracias a que al aplicar PCA sobre el vector concatenado solamente nos quedamos con las componentes que contienen la información relevante y se desechan las componentes que no contienen información útil.

(a) Secuencia de color.



(b) Secuencias segmentadas.



(c) Secuencias de profundidad.

Figura 3.7: Puntos característicos obtenidos mediante el algoritmo de Good Features to Track

```
%% Función para el tracking de los POI
%% Entrada: video - secuencia video; POI - puntos de interés a
   seguir
%% Salida: POI_Track - Trayectoria de los puntos seguidos.
function [POI_Track]=KLT_xy(POI, video);
        pointTracker= vision.PointTracker('MaxBidirectionalError',
           2); % Configuración del algoritmo KLT
        initialize(pointTracker, POI, objectFrame); %
           Inicialización del tracker
        for i=2:size(video,2) % Bucle para seguimiento de los
           puntos
             [Points] = step(pointTracker, frame); % Llamada al
                algoritmo KLT
             POI_Track=componente_XY(Points); % Función que nos
                devuelve las componentes x e y de las
                trayectorias obtenidas
        end
end
```

Las funciones de filtrado (*filter_ butterworth*) y análisis de componentes principales (*calc_ pca*) son las mismas que las usadas en el algoritmo base.

Se ha cambiado la función *calc_ DFT* [28] por la función *calc_ PSD*; este cambio se ha realizado ya que había numerosos trabajos que la usaban [24, 20, 25, 21]. Además, sus resultados son más discriminatorios, obteniendo en las pruebas realizadas un menor número de casos en los que no se ha escogido la frecuencia correcta. Con el uso de la DFT se han constatado situaciones en las que la frecuencia correcta era enmascarada en un armónico. Con este cambio se han mejorado los resultados, por lo que se ha implementado en el algoritmo propuesto.

```
%% Función el cálculo de la PSD
%% Entrada: PCA_results - Componentes principales; NFFT - número de
    puntos para el calculo de la PSD
%% Salida: PSD_results - Matriz con las Densidades Espectrales de
   Potencia de las 4 componentes
function [PSD_results]=calc_PSD(PCA_results, NFFT);
        for i=2:5
                PSD_results(i)=periodogram(PCA_results(i),NFFT)/(
                   NFFT-1); % Calculamos la PSD de las componentes
                    2 a 5
        end
```

```
8 end
```

En el caso de la obtención del ritmo cardíaco se ha procedido ha modificar la función (*find_HR*) para adaptar a los resultados de la PSD. La nueva función *find_HR_new* realiza la búsqueda de los máximos en cada una de las Densidades Espectrales de Potencia calculadas, para elegir la frecuencia a la que se produce el máximo absoluto de entre todos los máximos; una vez se ha obtenido esta frecuencia se multiplica por 60 segundos/minuto y se obtiene el ritmo cardíaco en pulsaciones por minuto.

```
1 %% Función nos devuelve el ritmo cardíaco
2 %% Entrada: PSD_results - PSD de las 4 componentes resultantes del
     PCA; NFFT - Número de puntos para el calculo de la PSD; fs -
     Frecuencia de muestreo
3 %% Salida: HR - ritmo cardíaco obtenido con el algoritmo
4 function [HR]=find_HR_new(PSD_results,NFFT,fs);
5         f_max=find_max(PSD_results, fs, NFFT); % Llamamos a la
              función que nos devuelve la frecuencia a la cual
              tenemos el máximo en la PSD, sus argumentos de entrada
              son las PSD calculadas, la frecuencia de muestreo y el
              número de puntos de la PSD.
6         HR=f_max*60; % Obtenemos el ritmo cardíaco multiplicando la
               frecuencia resultante por 60 segundos/minuto.
7 end
```

3.4. Desarrollos no incluidos

3.4.1. Introducción

En esta sección vamos a resumir los desarrollos que se han realizado pero que, o bien por su rendimiento inferior a los que finalmente se han incluido, o por la imposibilidad de llevarlos a cabo con los medios técnicos que disponíamos, se han descartado para el algoritmo propuesto.

Los desarrollos descartados son los siguientes:

- Uso de una única ROI para color.
- Uso de la variación en la componente z.
- Uso de otro tipos de puntos de interés (SIFT, SURF, BRISK).

- Conversión de coordenadas cartesianas a polares.

3.4.2. Desarrollo

El uso de una única ROI para el caso de color, se descartó ya que los resultados obtenidos en las pruebas que se realizaron con una única ROI reflejaban problemas de estabilidad. En el caso de que esta única ROI fuese toda la cabeza en secuencias de color, esta falta de estabilidad era producida por el parpadeo del individuo y por los movimientos producidos en la región de la boca, por ejemplo al tragar saliva o al hablar. En el caso de usar solamente una de las dos ROI, el problema de estabilidad venía producido por el bajo número de puntos de interés que se detectaban en ellas, el número de puntos se veía reducido casi a la mitad que con dos ROI.

Como ya se ha comentado en la sección del algoritmo propuesto, el uso de la variación en la componente z se descartó por la baja resolución de la primera versión de la cámara Kinect. Creemos que esta situación puede verse solventada con la nueva versión de la cámara Kinect, por ello este desarrollo no se descarta para un futuro.

Durante el desarrollo del trabajo se probaron diversas alternativas en cuanto al tipo de puntos de interés a usar: se realizaron pruebas con puntos SURF [32], SIFT [33] y BRISK [34], pero todas ellas nos arrojaban resultados peores que los que teníamos, lo que puede ser debido a que los puntos de interés que usamos actualmente son los puntos de interés recomendados para usar con el algoritmo KLT.

Por último, también se ha probado la conversión de las coordenadas cartesianas a polares. Estas pruebas se realizaron pensando que al haber una dependencia entre ambas coordenadas podrían proporcionarnos mejores resultados, pero al realizar los experimentos se vio que no aportaban ninguna mejoría en los resultados que se obtenían.

Capítulo 4

Evaluación

4.1. Introducción

En este capítulo se explica cómo se ha realizado y la composición del *dataset* que se ha elaborado para la realización de las pruebas. También expone el desarrollo de las pruebas que se han realizado para evaluar el funcionamiento de los algoritmos que se han desarrollado. Una vez se han obtenido los resultados de estas pruebas se ha procedido a analizarlos y a extraer las conclusiones que en este capítulo se exponen.

4.2. Marco de evaluación

4.2.1. *Dataset*

El *dataset* utilizado está compuesto por dos grandes bloques diferenciados: los vídeos que se han obtenido de la página web del proyecto del MIT[28] y las secuencias que se han grabado durante la realización del proyecto.

El conjunto de vídeos que proporciona el MIT es muy reducido y esto nos llevó a realizar nuestro propio conjunto de grabaciones.

El dataset, MMMDPRvds (Multimodal Multidistance Pulse Rate video dataset) que se ha generado está compuesto por secuencias de vídeo de 2 minutos de duración cada una de ellas.

Cada secuencia se ha gradado simultáneamente en 4 modalidades (tipos de secuencias): color procedente de webcam y profundidad, máscara segmentación y color, sincronizadas procedentes de Kinect; y además una secuencia de ground-truth, grabando el pulsímetro que llevaba el sujeto grabado (ver Figura 4.1). Para cada uno de los 5 tipos se han realizado 15 grabaciones a diferentes distancias, siendo estas distancias las 5 comprendidas entre 40 y 200 cm, ambas inclusive, con pasos de 40 cm

	Pulsaciones por minuto														
Distancia (cm)	40	80	120	160	200	220	240	260	280	300	320	340	360	380	400
1 s	66	69	67	67	66	68	73	68	64	70	71	67	69	73	70
2 s	66	69	67	68	66	68	73	69	63	72	71	68	68	73	69
3 s	66	67	67	68	66	68	71	70	63	73	70	69	68	72	69
4 s	66	67	67	69	66	67	70	70	63	73	68	70	67	70	68
5 s	66	68	67	70	66	66	68	70	64	73	67	72	67	69	67
6 s	66	68	68	70	66	66	67	70	65	73	66	72	66	69	65
7 s	66	68	68	71	66	65	67	70	67	73	65	71	67	67	64
8 s	65	68	68	71	66	65	66	70	68	72	64	69	68	66	63
9 s	64	68	69	72	68	65	65	70	70	71	64	67	69	66	63
10 s	64	67	69	72	64	64	65	70	70	70	64	65	69	65	63

Tabla 4.1: Ejemplo del Ground Truth

y las 10 de 220 a 400 cm, con incrementos de 20 cm. En todas las secuencias el sujeto se encuentra posando enfrentado a la cámara y en algunos momentos hablando.

Para la generación del ground-truth se ha anotado el pulso medio medido en cada segundo en las secuencias que contenían la grabación del pulsímetro. Para hacer uso de estos datos se han creado unas tablas que almacenan los datos del ritmo cardíaco organizados por sujeto, distancia a la cámara y cada segundo (30 *frames*), en la Tabla 4.1 podemos ver un ejemplo.

Aunque para todos los tipos de secuencias hay vídeos grabados en las 15 distancias, por motivos de funcionamiento de la cámara Kinect, las secuencias de profundidad no contienen información de la profundidad del sujeto hasta una distancia de 120 cm. Por este mismo motivo técnico del funcionamiento de Kinect, las secuencias con las máscaras de segmentación solamente contienen información en el rango comprendido entre 160 cm y 360 cm, en algunos casos podemos tener información de la segmentación en vídeos a 120 cm o 380 cm pero esta información es inestable para su análisis, por lo que se ha eliminado.

Estas grabaciones se han realizado con 3 voluntarios distintos y todas las grabaciones se han grabado en una única sesión para cada voluntario.

Los recursos materiales usados para las grabaciones han sido:

- Cámara Kinect v.1 con un frame rate de 30 frames/s y tamaño máximo de imagen de 640x480 para color, profundidad y segmentación, el tamaño de imagen usado para la grabación ha sido de 640x480.

- Webcam Logitech QuickCam Ultra Vision de 1.3 mega-pixeles de resolución, un tamaño máximo de imagen de 640x480 y frame rate de 30 frames/s, el tamaño

(a) RGB procedente de Kinect

(b) RGB procedente de webcam integrada

(c) Segmentación procedente de Kinect

(d) Profundidad procedente de Kinect

(e) RGB procedente de webcam externa

Figura 4.1: Muestras de los distintos tipos de secuencias que componen el dataset

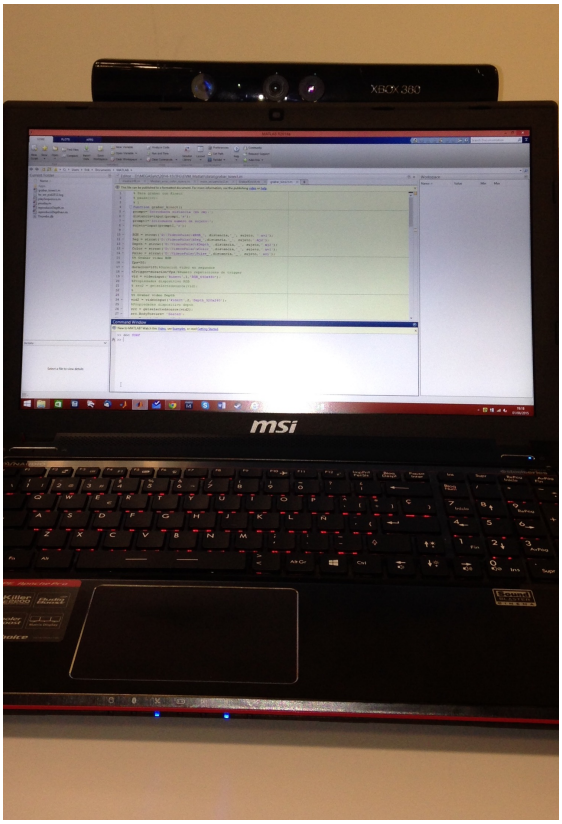


Figura 4.2: Posicionamiento webcam integrada y Kinect

de imagen usado para la grabación ha sido de 160x120. Con esta cámara se ha grabado el Pulsímetro Arimedical A-320.

- Webcam integrada en pc BisonCam NB Pro de 1.3 mega-pixeles de resolución, un tamaño máximo de imagen de 1280x960 y frame rate de 30 frames/s, el tamaño de imagen usado para la grabación ha sido de 640x360.

- PC MSI con procesador Intel core i-7 a 2.5 GHz, con 8 GB de memoria RAM y sistema operativo de 64 bits.

Para la grabación de las secuencias que componen el *dataset* se ha desarrollado un script, ver apéndice A. Este script permite grabar de forma sincronizada todas las fuentes de vídeo. Este hecho es importante ya que se necesita que todas las secuencias estén sincronizadas. En la Figura 4.2, se puede ver el posicionamiento de la webcam integrada y la cámara Kinect, podemos ver como se ha intentado que ambos sensores de color tuvieran una posición similar.

El script se ha implementado en Matlab, y nos permite mediante sus parámetros de entrada determinar la longitud de la secuencia a grabar, y dar nombre a los archivos

de salida generados. Los archivos de salida son: 2 vídeos en MJPEG2000 (profundidad y segmentación de Kinect) y 3 vídeos AVI (todas las secuencias de color).

4.2.2. Métricas

Para la evaluación de los algoritmos que se han desarrollado se han seleccionado las siguientes métricas usadas en [28].

- Media del pulso medido : Es la media del pulso medido en la duración de la secuencia, en pulsaciones por minuto.

$$Media\ pulso\ medido\ (ppm) = media(P,\ duración)$$

- Error: Es el error entre la media de la pulsación grabada del pulsímetro en la duración de la secuencia y la media del pulso arrojado por el algoritmo en la duración de la secuencia.

$$Error(\%) = |\frac{Media\ pulso_{Ground\ Truth}(P,\ duración) - Media\ pulso_{medido}(P,\ duración)}{Media\ pulso_{Ground\ Truth}(P,\ duración)}|x100$$

Durante el desarrollo del trabajo también se han usado otras métricas, pero solamente se usarán las anteriores al ser las propuestas en [28].

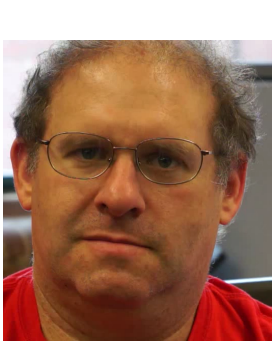
4.3. Pruebas y resultados

4.3.1. Pruebas y resultados del algoritmo base

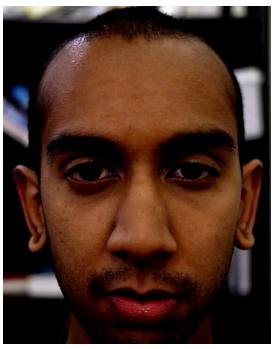
El objetivo de este primer conjunto de pruebas con los vídeos utilizados por el artículo original del MIT, era corroborar que con nuestro algoritmo base se obtenían los mismos resultados que con el algoritmo desarrollado por el MIT. En estas pruebas se han comparado los resultados publicados en [28] con los obtenidos con nuestra implementación.

Para estas pruebas solamente hemos podido acceder a 3 de los vídeos usados por el MIT, disponibles en [2], ver Figura 4.3. Dos de ellos son secuencias colaborativas del sujeto en posición frontal a una distancia reducida respecto de la cámara, con una duración de 10 segundos. El tercero de los vídeos es una grabación de un bebé a una mayor distancia y con una posición frontolateral de la región facial y una duración de 30 segundos.

Como se puede ver en la Tabla 4.2, los resultados que se han obtenido con el algoritmo base son muy próximos a los resultados presentados en el artículo[28]. Se puede observar que el error entre los resultados del algoritmo del MIT y nuestra implementación del algoritmo base es inferior al 2.3 % en los tres casos bajo estudio.



(a) face_54bpm



(b) face_55bpm

(c) baby_152bpm

Figura 4.3: Vídeos MIT. Fuente: [2]

	Media del pulso (ppm)				
Vídeo	Ground-truth	Algoritmo MIT datos de [28] (% error)	Algoritmo base (% error)	Algoritmo propuesto (% error)	Error (%) MIT-base
face_54bpm	54	**55 (1.85)**	56.25 (4.17)	52.29 (-3.17)	2.27
face_55bpm	55	56 (1.78)	**54.75 (-0.46)**	53.50 (-2.73)	2.23
baby_152bpm	152	**152 (0)**	**152 (0)**	**152 (0)**	0

Tabla 4.2: Resultados algoritmo base con dataset MIT

Estos resultados se pueden interpretar como que nuestro algoritmo base funciona de forma muy similar al algoritmo propuesto por el MIT.

4.3.2. Pruebas y Resultados del algoritmo propuesto

A la vista de los resultados del apartado anterior se puede determinar que el algoritmo base es una buena aproximación al algoritmo del MIT. En este apartado presentaremos una evaluación comparativa del algoritmo base con la implementación propuesta, mediante la realización de una batería de pruebas haciendo uso del MMMDPRvds desarrollado.

Estas pruebas han consistido en realizar ejecuciones en las que se ha ido variando la duración del enventanado usado, variando desde ventanas de 2 segundos hasta ventanas de 11 segundos y el uso de los algoritmo tanto haciendo uso de la DFT como de la PSD.

De esta forma se han obtenido una gran cantidad de datos, pudiendo así seleccionar

el tamaño de ventana más adecuado para cada algoritmo. También se ha podido analizar el comportamiento de los distintos algoritmos en función de la distancia al sujeto.

En las siguientes secciones se presentan los resultados seleccionados, por simplicidad no se mostrarán todos los casos probados, que han dado lugar a la elección de parámetros "por defecto", para finalmente analizar los resultados con esos valores.

4.3.2.1. Selección del tamaño de ventana

En estas pruebas hemos aplicado el algoritmo sobre la totalidad del MMMDPRvds, probando los tamaños de ventada comprendidos entre 2 segundos y 11 segundos tanto para nuestro algoritmo base como para el algoritmo propuesto.

Podemos observar en la Tabla4.4 como las ventanas de tamaño superior tienen un mejor comportamiento. Este comportamiento es general para todas las distancias que se han analizado.

En la Tabla 4.3 se puede ver un ejemplo de comparativa de los resultados de los diferentes algoritmos en función del tamaño de ventana. Se han obtenido resultados para las todos los tipos de secuencias grabadas en todas las distancias.

A la vista de los resultados 4.4 arrojados por estas pruebas, se ha decidido usar el tamaño de ventana de 11 segundos para las pruebas siguientes. También podemos ver como los resultados sobre las secuencias grabadas con la webcam son mejores y esto se debe a la diferente calidad de imagen.

4.3.2.2. Selección de la técnica de dominio transformado

Para estas pruebas se han procesado todos los vídeos del MMMDPRvds fijando el tamaño de la ventana en 11 segundos. En ellas se ha ejecutado nuestro algoritmo propuesto obteniendo resultados tanto para el calculo del ritmo cardíaco usando la FFT y como usando la DFT.

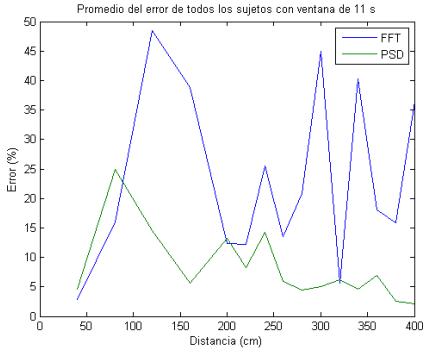
Como podemos ver en la Figura 4.4, el funcionamiento de los distintos algoritmos propuestos, es en general mejor haciendo uso de la densidad espectral de potencia en lugar de la Transformada Discreta de Fourier.

	Error (%)									
Algoritmo	Base		Propuesto PSD				Propuesto DFT			
Tamaño de ventana (s)	Webcam	Color Kinect	Webcam	Color Kinect	Segmentación	Profundidad	Webcam	Color Kinect	Segmentación	Profundidad
2	56.76	65.18	25.93	30.76	13.86	60.93	53.69	79.44	49.67	99.89
3	50.73	43.43	11.38	16.05	9.56	33.50	47.09	45.27	46.08	99.59
4	44.72	32.87	4.36	12.75	8.87	30.15	46.72	39.56	45.09	95.24
5	41.64	28.47	1.13	10.12	8.68	26.32	35.71	25.19	39.43	94.13
6	34.77	30.64	0.68	8.59	10.78	25.66	34.72	25.46	40.55	88.65
7	31.22	20.05	0.52	6.08	8.60	27.83	28.67	12.91	42.95	89.06
8	25.01	23.84	1.31	5.12	9.47	25.05	25.05	18.69	38.82	87.04
9	14.94	30.77	1.95	8.74	3.72	23.50	17.24	12.43	47.02	90.64
10	13.85	14.76	0.24	5.79	9.73	20.85	10.54	8.70	40.69	94.00
11	13.47	37.46	1.08	5.51	7.64	23.75	8.45	8.56	41.61	90.55

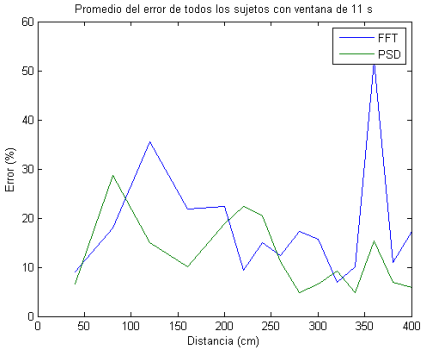
Tabla 4.3: Comparativa entre ventanas para una distancia fija de 320 cm sobre el secuencias del sujeto 2

Error (%)										
Algoritmo	Base		Propuesto PSD				Propuesto DFT			
Tamaño de ventana (s)	Webcam	Color Kinect	Webcam	Color Kinect	Segmentación	Profundidad	Webcam	Color Kinect	Segmentación	Profundidad
2	75.09	78.82	37.05	35.85	25.37	64.01	69.21	77.83	63.69	91.49
3	63.61	70.23	21.86	20.52	21.43	49.43	58.59	61.95	57.55	73.05
4	58.67	63.23	15.40	14.15	21.12	43.06	53.01	48.84	54.94	65.26
5	54.03	60.54	12.43	13.30	20.41	38.24	45.15	39.49	53.47	57.35
6	56.33	59.42	10.83	12.41	20.36	36.56	40.43	32.93	52.39	54.03
7	57.71	58.75	9.01	12.50	19.13	35.24	36.80	28.27	51.56	51.77
8	56.62	59.53	9.20	**12.29**	17.75	33.44	33.84	26.09	51.20	49.81
9	57.39	59.36	8.81	12.45	17.40	31.77	28.87	21.48	**50.76**	45.65
10	57.86	58.33	8.51	12.61	16.66	**31.15**	25.83	20.85	50.80	44.78
11	**28.61**	**31.81**	**8.23**	12.44	**16.47**	31.47	**23.46**	**18.26**	50.80	**43.13**

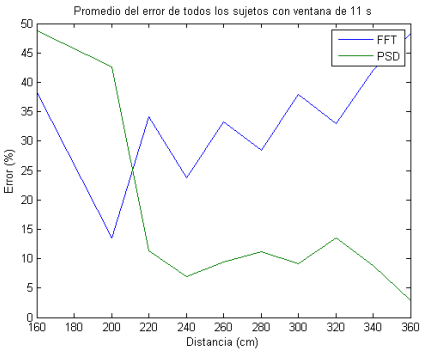
Tabla 4.4: Media del error en todas las distancias en función del tamaño de la ventana



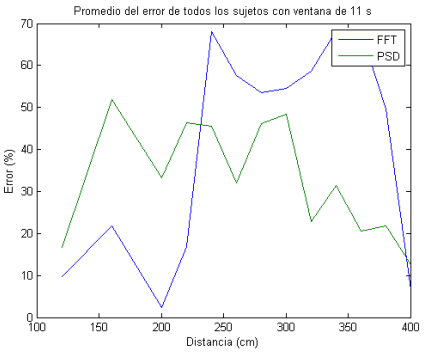
(a) Algoritmo propuesto con secuencia de webcam



(b) Algoritmo propuesto con secuencia de color Kinect



(c) Algoritmo propuesto con secuencias de segmentación



(d) Algoritmo propuesto con secuencias de profundidad

Figura 4.4: Comparativa del uso de la DFT y PSD en los distintos algoritmos

	Error (%)					
Algoritmo	Base		Propuesto			
Distancia (cm)	Webcam	Color Kinect	Webcam	Color Kinect	Segmentación	Profundidad
40	7.56	29.36	**4.59**	6.59	-	-
80	33.44	**14.18**	24.92	28.62	-	-
120	28.72	24.08	**14.50**	14.93	-	16.72
160	26.40	25.81	**5.65**	10.06	48.76	51.85
200	25.29	21.40	**13.24**	18.92	42.68	33.33
220	25.69	27.40	**8.29**	22.30	11.39	46.29
240	22.50	22.50	14.26	20.41	**6.97**	45.59
260	22.34	22.34	**5.90**	11.13	9.47	32.12
280	25.49	24.69	**4.47**	4.81	11.14	46.22
300	29.12	19.16	**5.12**	6.57	9.10	48.38
320	27.53	24.33	**6.26**	9.29	13.49	22.76
340	24.86	20.71	**4.56**	4.81	8.85	31.42
360	31.35	30.73	6.97	15.42	**2.82**	20.65
380	44.42	91.33	**2.59**	6.87	-	21.71
400	54.46	79.17	**2.19**	5.86	-	12.61

Tabla 4.5: Media del error en función de la distancia con ventana de 11 segundos y PSD

4.3.2.3. Análisis en función de la distancia

Para realizar el análisis en función de la distancia se han ejecutado los algoritmos base y propuesto haciendo uso de todas las secuencias del MMMDPRvds, con la ventana fijada en un tamaño de 11 segundos y el uso de PSD.

A la vista de los resultados, que se pueden ver en la Tabla 4.5 y en la Figura 4.5, podemos ver el comportamiento de los distintos algoritmos en función de la distancia a la que se encuentra el sujeto. Se puede observar como hay un buen comportamiento del algoritmo de color en todo el rango de distancias con un error medio del 8.23 % en el *dataset*. En el caso de 80 cm hay movimiento voluntario de la cabeza del sujeto en 2 de las 3 secuencias grabadas a esta distancia y esto nos provoca un mayor error. También vemos que el algoritmo funciona mejor sobre las secuencias obtenidas por la webcam frente a las obtenidas por la cámara Kinect, y esto debido a la diferencia de calidad de las imágenes. El algoritmo propuesto con secuencias de segmentación tiene una comportamiento bastante estable con un error entre el 13.49 % y el 2.82 %, en el rango comprendido entre 220 cm y 360 cm. También se ha observado que el algoritmo propuesto con secuencias de profundidad tiene unos resultados peores que el resto de algoritmos.

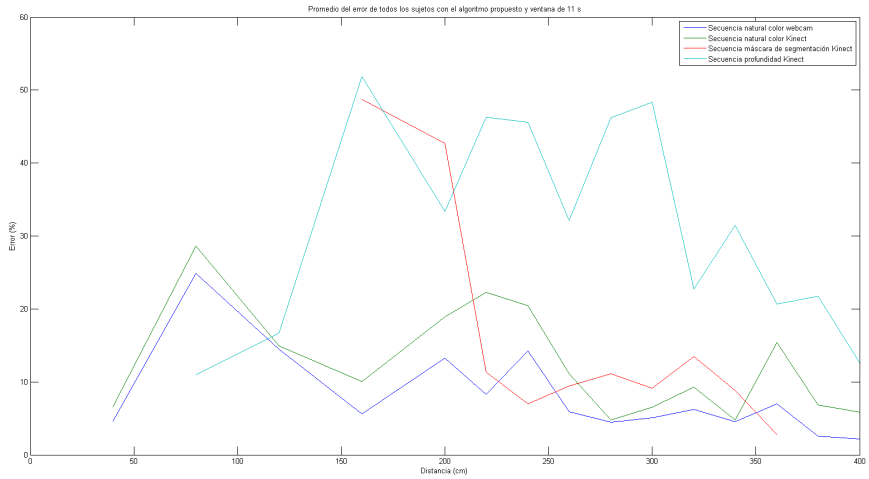


Figura 4.5: Gráfica comparativa algoritmos en función de la distancia con ventana de 11 segundos y PSD

4.4. Conclusión

Podemos concluir que nuestro trabajo ha conseguido tener unos resultados similares, para los tres vídeos de los que disponíamos, entre los descritos en el artículo que presentaba el algoritmo original del MIT [28] y el algoritmo base desarrollado.

También se puede afirmar que nuestra propuesta con secuencias de color tiene un comportamiento estable y bastante preciso en todos los casos bajo estudio, mejorando los resultados del algoritmo base en estas mismas condiciones.

En lo referente máscara de segmentación, se ha podido determinar que tiene un buen funcionamiento cuando nos encontramos en un rango de distancias de 220 cm a 400 cm.

Por último vemos que el comportamiento del algoritmo sobre las secuencias de profundidad es peor que con el resto de modalidades.

A la vista de los resultados de las pruebas realizadas podemos determinar que la mejora en los resultados es debida principalmente al uso de la Densidad Espectral de Potencia, y el resto de cambios suponen una mejora marginal.

Capítulo 5

Conclusiones y trabajo futuro

5.1. Conclusiones

El objetivo principal de este trabajo era desarrollar un algoritmo capaz de medir el ritmo cardíaco de una persona mediante los movimientos imperceptibles que son inducidos por el flujo sanguíneo en la cabeza. Después del análisis realizado en este documento de las distintas implementaciones realizadas, podemos concluir que el objetivo se ha cumplido de forma razonable. También podemos decir que se han cumplido los objetivos de adaptación del algoritmo a secuencias de profundidad: aunque su rendimiento es inferior al resto de modalidades como hemos podido ver en los resultados hay resultados en los que podemos apreciar el posible potencial, y por ello se piensa que podría mejorar con las propuestas que mencionaremos en la siguiente sección.

Otro de los objetivos de este trabajo era la realización de un *dataset* que sirviese para la evaluación metodología de los distintos algoritmos; como se ha mencionado en el documento ese *dataset* se ha realizado y será usado para posibles trabajos derivados de éste.

También se ha analizado la viabilidad de un posible sistema combinado en función de la distancia. Una vez analizado el comportamiento del algoritmo propuesto con las distintas modalidades hemos visto que la modalidad de color tiene un mejor comportamiento en la práctica totalidad de las distancias, por lo tanto con la implementación actual queda descartado un sistema combinado.

5.2. Trabajo futuro

A la vista de los resultados que se han obtenido en este trabajo se propone trabajar en:

- Profundizar en el trabajo con cámaras de profundidad. Para ello se recomienda el uso de la nueva cámara Kinect 2, ya que se piensa que la mejor resolución de esta en las secuencias de profundidad podría proporcionar una sustancial mejora de los resultados de este algoritmo. Sobre este algoritmo también cabe mencionar que se debería trabajar en la detección automática de la región de interés, en la posible mejora de los puntos de interés obtenidos para generar las trayectorias de movimiento, y en la explotación del movimiento en la coordenada z del espacio.

- Implementar un sistema combinado que funcione con el algoritmo sobre secuencias de color, máscaras de segmentación y de profundidad en función de la distancia del sujeto a la cámara y los resultados que se han obtenido en las pruebas de estos.

- Implementar estas aproximaciones dentro de una aplicación en tiempo real, para lo que se debería implementar el sistema en un lenguaje de alto nivel, como podría ser C++ con el uso de las librerías de OpenCV, y desarrollar una interfaz de usuario.

- Ampliar el dataset con secuencias del mismo sujeto y realizadas en la misma sesión con diferencias notables del ritmo cardíaco, con el fin de poder evaluar si el error se mantiene constante en los cambios de ritmo cardíaco.

- Proponer nuevas métricas para la evaluación del algoritmo, ya que creemos que las actuales no evalúan de forma completa el funcionamiento del algoritmo.

Bibliografía

[1] M. Z. Poh, D. J. McDuff, and R. W. Picard, "Advancements in noncontact, multiparameter physiological measurements using a webcam," *IEEE Transactions on Biomedical Engineering*, vol. 58, no. 1, pp. 7–11, 2011.

[2] M. Rubinstein, "http://people.csail.mit.edu/mrub/evm/," 2012.

[3] H.-Y. Wu, M. Rubinstein, E. Shih, J. Guttag, F. Durand, and W. Freeman, "Eulerian video magnification for revealing subtle changes in the world," *ACM Transactions on Graphics*, vol. 31, no. 4, pp. 1–8, 2012.

[4] D. He, E. S. Winokur, and C. G. Sodini, "A continuous, wearable, and wireless heart monitor using head ballistocardiogram (bcg) and head electrocardiogram (ecg)," in *Engineering in Medicine and Biology Society, EMBC, 2011 Annual International Conference of the IEEE*, pp. 4729–4732, IEEE, 2011.

[5] I. Starr, A. Rawson, H. Schroeder, and N. Joseph, "Studies on the estimation of cardiac ouptut in man, and of abnormalities in cardiac function, from the heart's recoil and the blood's impacts; the ballistocardiogram," *American Journal of Physiology–Legacy Content*, vol. 127, no. 1, pp. 1–28, 1939.

[6] J. L. I. Pavlidis, "Thermal image analysis for polygraph testing," *IEEE Engineering in Medicine and Biology Magazine*, vol. 21, pp. 56–64, 2002.

[7] C. Takano and Y. Ohta, "Heart rate measurement based on a time-lapse image.," *Medical engineering & physics*, vol. 29, pp. 853–857, Oct. 2007.

[8] Philips, "Philips vitals signs camera, "http://www.vitalsignscamera.com"," 2011.

[9] C. Inc., "Cardiio app, "http://www.cardiio.com/about"," 2013.

[10] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, pp. 511–518, 2001.

[11] T. Fitzpatrick, "Sun and skin (soleil et peau)," *J Med Esthétique*, vol. 2, pp. 33–34, 1975.

[12] F. Bousefsaf, C. Maaoui, and A. Pruski, "Continuous wavelet filtering on webcam photoplethysmographic signals to remotely assess the instantaneous heart rate," *Biomedical Signal Processing and Control*, vol. 8, pp. 568–574, Nov. 2013.

[13] A. Kleiner and D. Rabinowitz, "Video Heart Rate Detection," 2014.

[14] Y. Hsu, Y. L. Lin, and W. Hsu, "Learning-based heart rate detection from remote photoplethysmography features," *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, pp. 4433–4437, 2014.

[15] S. Kwon, H. Kim, and K. S. Park, "Validation of heart rate extraction using video imaging on a built-in camera system of a smartphone," *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS*, pp. 2174–2177, 2012.

[16] M. Lewandowska, J. Ruminski, T. Kocejko, and J. Nowak, "Measuring pulse rate with a webcam - a non-contact method for evaluating cardiac activity," in *Computer Science and Information Systems (FedCSIS), 2011 Federated Conference on*, pp. 405–410, IEEE, 2011.

[17] M.-Z. Poh, D. J. McDuff, and R. W. Picard, "Non-contact, automated cardiac pulse measurements using video imaging and blind source separation.," *Optics express*, vol. 18, no. 10, pp. 10762–10774, 2010.

[18] M.-Z. Poh, D. McDuff, and R. Picard, "A medical mirror for non-contact health monitoring," in *ACM SIGGRAPH 2011 Emerging Technologies*, p. 2, ACM, 2011.

[19] T. Pursche, J. Krajewski, and R. Moeller, "Video-based heart rate measurement from human faces," in *Consumer Electronics (ICCE), 2012 IEEE International Conference on*, pp. 544–545, IEEE, 2012.

[20] L. Scalise, N. Bernacchia, I. Ercoli, and P. Marchionni, "Heart rate measurement in neonatal patients using a webcamera," *MeMeA 2012 - 2012 IEEE Symposium on Medical Measurements and Applications, Proceedings*, pp. 6–9, May 2012.

[21] G. R. Tsouri, S. Kyal, S. Dianat, and L. K. Mestha, "Constrained independent component analysis approach to nonobtrusive pulse rate measurements," *Journal of biomedical optics*, vol. 17, no. 7, pp. 0770111–0770114, 2012.

[22] W. Verkruysse, L. O. Svaasand, and J. S. Nelson, "Remote plethysmographic imaging using ambient light," *Optics express*, vol. 16, no. 26, pp. 21434–21445, 2008.

[23] L. Wei, Y. Tian, Y. Wang, T. Ebrahimi, and T. Huang, "Automatic webcam-based human heart rate measurements using laplacian eigenmap," in *Computer Vision–ACCV 2012*, pp. 281–292, Springer, 2013.

[24] J. B. Bolkhovsky, C. G. Scully, and K. H. Chon, "Statistical analysis of heart rate and heart rate variability monitoring through the use of smart phone cameras," in *Engineering in Medicine and Biology Society (EMBC), 2012 Annual International Conference of the IEEE*, pp. 1610–1613, IEEE, 2012.

[25] C. Scully, J. Lee, J. Meyer, A. M. Gorbach, D. Granquist-Fraser, Y. Mendelson, and K. H. Chon, "Physiological parameter monitoring from optical recordings with a mobile

phone," *Biomedical Engineering, IEEE Transactions on*, vol. 59, no. 2, pp. 303–306, 2012.

[26] L. Tarassenko, M. Villarroel, A. Guazzi, J. Jorge, D. Clifton, and C. Pugh, "Non-contact video-based vital sign monitoring using ambient light and auto-regressive models," *Physiological measurement*, vol. 35, no. 5, p. 807, 2014.

[27] L. Carvalho, M. Virani, and M. Kutty, "Analysis of Heart Rate Monitoring Using a Webcam," *Analysis*, vol. 3, no. 5, pp. 6593–6595, 2014.

[28] G. Balakrishnan, F. Durand, and J. Guttag, "Detecting pulse from head motions in video," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 3430–3437, June 2013.

[29] J. Shi and C. Tomasi, "Good features to track," in *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*, pp. 593–600, IEEE, 1994.

[30] B. D. Lucas, T. Kanade, *et al.*, "An iterative image registration technique with an application to stereo vision.," in *IJCAI*, vol. 81, pp. 674–679, 1981.

[31] R. Irani, K. Nasrollahi, and T. B. Moeslund, "Improved Pulse Detection from Head Motions Using DCT," *International Conference on Computer Vision Theory and Applications*, p. 8, 2014.

[32] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *Computer vision–ECCV 2006*, pp. 404–417, Springer, 2006.

[33] D. Lowe, "Object recognition from local scale-invariant features," *The Proceedings of the Seventh IEEE International Conference on Computer Vision*, vol. 2, pp. 1150 – 1157, 1999.

[34] M. C. Leutenegger, S. and R. Siegwart., "Brisk: Binary robust invariant scalable keypoints," in *Proceedings of the IEEE International Conference, ICCV*, 2011.

Apéndice A

Script para grabación del dataset

```
%%Función para la grabación del dataset
function grabar_dataset()
        prompt='Introduzca distancia (en cm):'; % Se pide la
           distancia a la que estamos grabando
        distancia=input(prompt,'s');
        prompt1='Introduzca número de sujeto:'; % Se pide el número
            de sujeto
        sujeto=input(prompt1,'s');
        prompt2='Introduzca duración (s):'; % Se pide la duración
           en segundos
        duration=input(prompt2);
        % Se generan los nombres de los archivos de salida
        RGB=strcat('D:\VideosPulso\kRGB_', distancia,'_', sujeto, '
           .avi');
        Seg=strcat('D:\VideosPulso\kSeg_',distancia,'_', sujeto,'.
           mj2');
        Depth=strcat('D:\VideosPulso\kDepth_',distancia,'_', sujeto
           ,'.mj2');
        Color=strcat('D:\VideosPulso\wColor_',distancia,'_', sujeto
           ,'.avi');
        Pulso=strcat('D:\VideosPulso\Pulse_',distancia,'_', sujeto,
           '.avi');

        %% Parámetros para la grabación
        fps=30;
        nTrigger=duracion*fps; %Numero repeticiones de trigger
        % Fuente de entrada Kinect Color
        vid=videoinput('kinect',1,'RGB_640x480');
```

```
21
22          % Fuente de entrada Kinect depth
23          vid2=videoinput('kinect',2,'Depth_320x240');
24
25          %Propiedades del dispositivo depth
26          src=getselectedsource(vid2);
27          src.BodyPosture='Seated';
28          src.TrackingMode='Skeleton';
29          src.SkeletonsToTrack=1;
30
31          % Fuente de entrada webcam integrada
32          vid3=videoinput('winvideo', 2, 'RGB24_160x120');
33          src3=getselectedsource(vid3);
34
35          % Fuente de entrada webcam externa
36          vid4=videoinput('winvideo', 1, 'MJPG_640x360');
37          src4=getselectedsource(vid4);
38
39          %% Configuramos los parámetros para la grabación
40          vid.FramesPerTrigger = 1;
41          vid2.FramesPerTrigger = 1;
42          vid3.FramesPerTrigger=1;
43          vid4.FramesPerTrigger = 1;
44
45          vid.TriggerRepeat = nTrigger;
46          vid2.TriggerRepeat = nTrigger;
47          vid3.TriggerRepeat = nTrigger;
48          vid4.TriggerRepeat = nTrigger;
49
50          % Establecemos el trigger de forma manual para sincronizar
               las 4 fuentes
51          triggerconfig([vid vid2 vid3 vid4],'manual');
52
53          %% Establecemos los parámetros de los flujos de datos para
               grabar en disco
54          diskLoggerKRGB=VideoWriter(RGB, 'Uncompressed AVI');
55          diskLoggerKRGB.FrameRate=fps;
56          diskLoggerkSeg=VideoWriter(Seg, 'Archival');
57          diskLoggerkSeg.FrameRate=fps;
58          diskLoggerkSeg.LosslessCompression=false;
59          diskLoggerkSeg.MJ2BitDepth=8;
60          diskLoggerkDepth=VideoWriter(Depth, 'Motion JPEG 2000');
```

```
61          diskLoggerkDepth.FrameRate=fps;
62          diskLoggerkDepth.MJ2BitDepth=16;
63          diskLoggerWColor=VideoWriter(Color, 'Uncompressed AVI');
64          diskLoggerWColor.FrameRate=fps;
65          diskLoggerPulse=VideoWriter(Pulso, 'Uncompressed AVI');
66          diskLoggerPulse.FrameRate=fps;
67
68          %%  Creamos los flujos de datos para grabar en disco
69          open(diskLoggerKRGB);
70          open(diskLoggerkDepth);
71          open(diskLoggerkSeg);
72          open(diskLoggerWColor);
73          open(diskLoggerPulse);
74
75          %% Comenzamos la grabacción
76          start([vid vid2 vid3 vid4]);
77
78          for i = 1:nTrigger
79                  trigger([vid vid2 vid3 vid4])
80                  [imgColor]=getdata(vid);
81                  [imgDepth, ~, metaData_Depth]=getdata(vid2);
82                  imgSeg=uint8(metaData_Depth.SegmentationData);
83                  writeVideo(diskLoggerKRGB, imgColor);
84                  writeVideo(diskLoggerkDepth, imgDepth);
85                  writeVideo(diskLoggerkSeg, imgSeg);
86                  writeVideo(diskLoggerWColor, getdata(vid4));
87                  writeVideo(diskLoggerPulse, getdata(vid3));
88          end
89
90          %% Finalizamos la grabación
91          stop([vid vid2 vid3 vid4]);
92
93          %% Cerramos los flujos de datos
94          close(diskLoggerKRGB);
95          close(diskLoggerkSeg);
96          close(diskLoggerkDepth);
97          close(diskLoggerWColor);
98          close(diskLoggerPulse);
99  end
```

Apéndice B

Interfaz para visualización del resultados

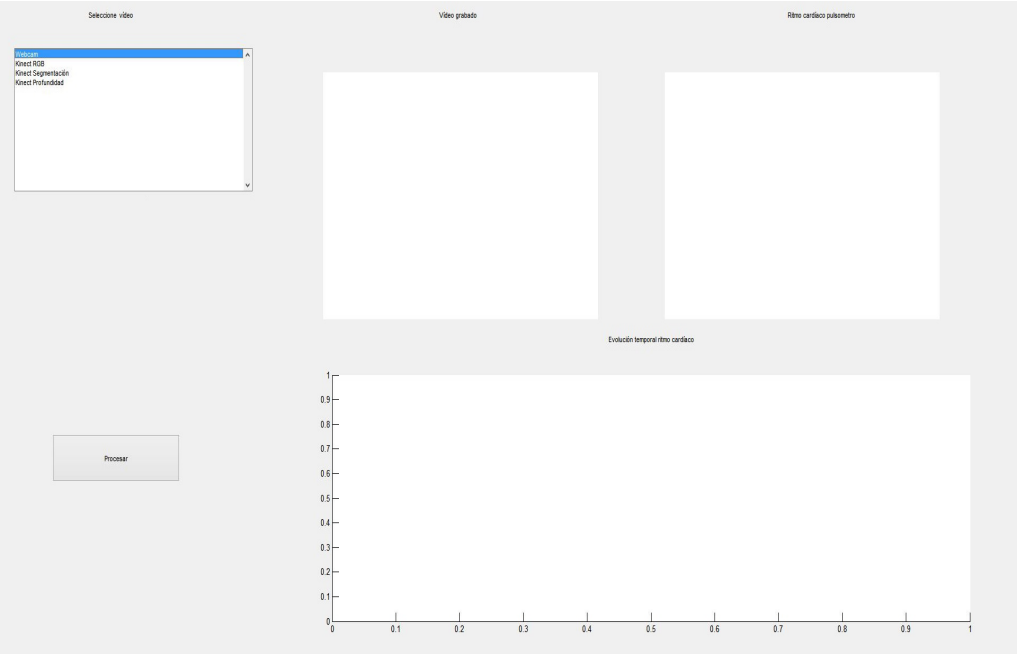


Figura B.1: Interfaz para la visualización de resultados

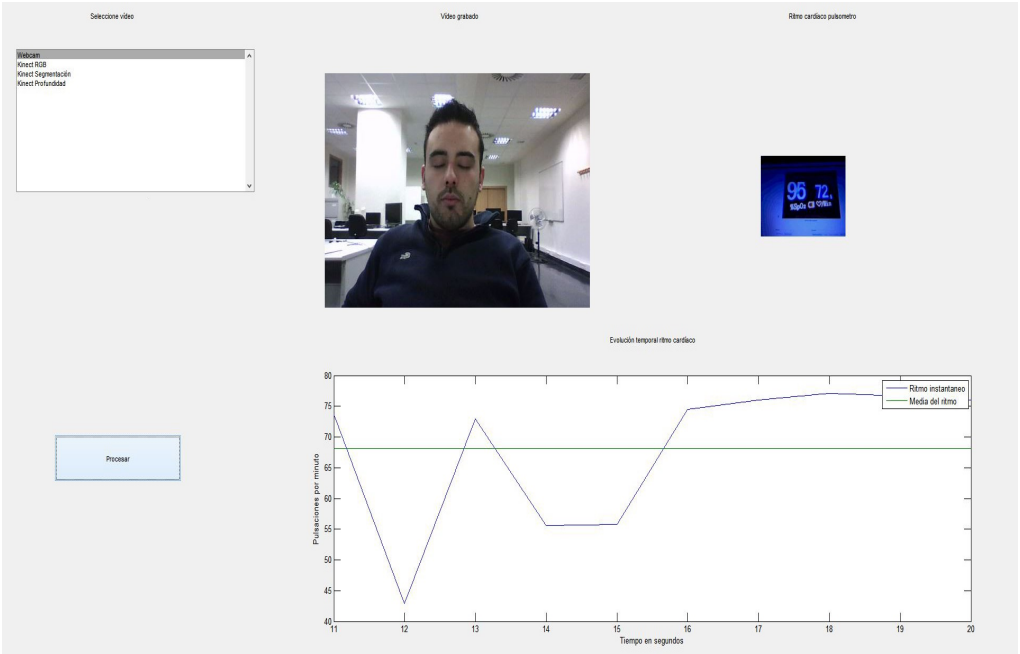


Figura B.2: Ejemplo de uso de la interfaz

Apéndice C

Interfaz para demostración

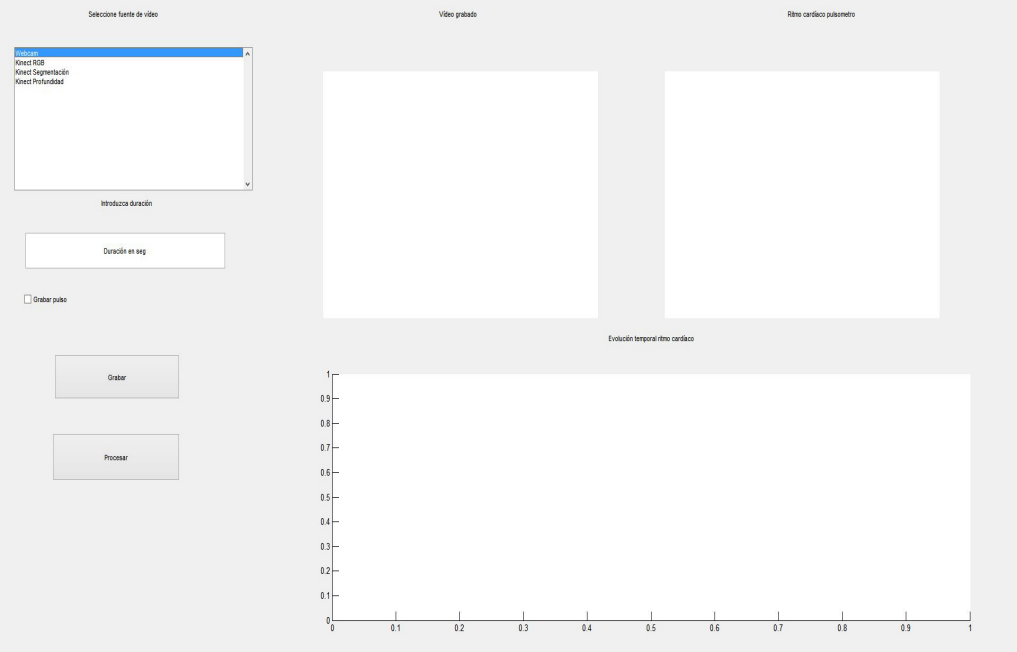


Figura C.1: Interfaz para demostración